

Toward Efficient Deep Spiking Neuron Networks: A Survey On Compression

Hui Xie¹, Ge Yang¹, and Wenjuan Gao¹

Beihang University, Beijing, CHINA

{xiehui, 20231102, wjgao}@buaa.edu.cn

Abstract. With the rapid development of deep learning, Deep Spiking Neural Networks (DSNNs) have emerged as promising due to their unique spike event processing and asynchronous computation. When deployed on neuromorphic chips, DSNNs offer significant power advantages over Deep Artificial Neural Networks (DANNs) and eliminate time and energy consuming multiplications due to the binary nature of spikes (0 or 1). Additionally, DSNNs excel in processing temporal information, making them potentially superior for handling temporal data compared to DANNs. However, their deep network structure and numerous parameters result in high computational costs and energy consumption, limiting real-life deployment. To enhance DSNNs efficiency, researchers have adapted methods from DANNs, such as pruning, quantization, and knowledge distillation, and developed specific techniques like reducing spike firing and pruning time steps. While previous surveys have covered DSNNs algorithms, hardware deployment, and general overviews, focused research on DSNNs compression and efficiency has been lacking. This survey addresses this gap by concentrating on efficient DSNNs and their compression methods. It begins with an exploration of DSNNs' biological background and computational units, highlighting differences from DANNs. It then delves into various compression methods, including pruning, quantization, knowledge distillation, and reducing spike firing, and concludes with suggestions for future research directions.

Keywords: Deep Spiking Neuron Networks · Pruning · Quantization · Knowledge distillation · Reducing Spiking Firing · Compression

1 Introduction

As early as 1997, Maass [52] classified Spiking Neural Networks (SNNs) as third-generation neural networks based on their computational units, specifically spiking neurons or "integrate-and-fire neurons". In contrast, first-generation neural networks, developed around the 1950s and known as perceptrons, were based on McCulloch-Pitts neurons. These networks struggled with problems that were not linearly separable, such as the XOR operation. To address these limitations, second-generation neural networks, or Artificial Neural Networks (ANNs), were developed, utilizing activation functions that apply a continuous set of output values to a weighted sum (or polynomial) of the inputs.

With the advancement of neuromorphic computing [61], SNNs have demonstrated significant potential due to their unique spike event processing and asynchronous computation capabilities. When deployed on neuromorphic chips, SNNs offer substantial

power advantages over ANNs and avoid time and energy consuming multiplication operations due to the binary nature of spikes (0 or 1) [3] . Moreover, SNNs more closely mimic the complex workings of the biological brain [34], enhancing their potential to achieve genuine artificial intelligence . Their ability to process temporal information naturally makes them particularly suited for tasks involving temporal data [45], surpassing the capabilities of ANNs in this regard . SNNs are also uniquely suited for applications like event-based cameras [20], which are event-driven, asynchronous, and binary, making them ideal for processing with SNNs on neuromorphic chips .

In recent years, deep learning [40] has achieved tremendous success with DANNs [6,84,53,78] and, correspondingly, with DSNNs [36,83,49]. Numerous new architectures have been proposed to improve performance on various tasks in DANNs [37,68,29,33] , with similar advancements seen in DSNNs, including Spiking VGG [41,64] , Spiking ResNet [32,19] , and Spiking Transformers [82,67,79] .

However, the deep network structure and large number of parameters in these models lead to significant computational costs and energy consumption, limiting their practical deployment. To achieve efficient DSNNs, researchers have adapted methods from DANNs, such as pruning, quantization, and knowledge distillation. Additionally, unique methods specific to DSNNs, such as reducing spike firing and pruning time steps, have been developed.

Previous surveys on SNNs have primarily focused on algorithms [12,81] , hardware deployment [4] , and general overviews [60] , without specifically addressing DSNNs compression and efficiency. Conversely, there is extensive research on compressing DANNs, including structured pruning [30], quantization [22], knowledge distillation [23] . Therefore, a focused survey on DSNNs compression is needed.

This survey addresses this gap by concentrating on efficient DSNNs and their compression methods. It begins with an exploration of the biological background and computational units of SNNs to understand how they differ from ANNs. It then analyzes various compression methods, including pruning, quantization, knowledge distillation, and reducing spike firing. Finally, it suggests directions for future research.

2 Background

The introduction of activation functions in neural networks is primarily driven by the need to introduce non-linear functions into a linear system, thereby increasing its complexity and enhancing its representative capabilities. In contrast, the computational units in SNNs, such as the Leaky Integrate-and-Fire (LIF) neuron or Integrate-and-Fire (IF) neuron, are designed to more closely mimic the properties of biological neurons. This distinction in computational units is a fundamental difference between SNNs and ANNs.

2.1 Biological Background

A typical neuron consists of three main parts: dendrites, soma, and axon. Dendrites collect input signals from other neurons and transmit them to the soma. The soma acts as a

computational unit, generating an action potential when the accumulated incoming current causes the membrane potential to exceed a certain threshold. This action potential then travels along the axon and transmits the signal to the next neuron through synapses at the axon's terminal.

LIF Model. The Leaky Integrate-and-Fire (LIF) model, first proposed by Lapicque in 1907, describes the process of action potentials in neurons. Neurons fire impulses when the membrane potential reaches the threshold voltage $V_{threshold}$, after which the membrane potential resets to the resting potential V_{reset} . The LIF model focuses on the patterns of sub-threshold potential voltage variations. [14].

$$\tau_m \frac{dV}{dt} = V_{reset} - V + R_m I. \quad (1)$$

where τ_m represents the membrane time constant, V_{reset} is the resting potential, and R_m and I are the cell membrane's impedance and the input current, respectively.

The LIF neuron model provides a simplified representation of biological neurons, emphasizing essential features like membrane potential leakage, accumulation, and excitation. While its biological accuracy is limited, its simplicity makes it suitable for computational simulations.

Other Models. The Hodgkin-Huxley (H-H) model, proposed by Hodgkin and Huxley in 1952 [31], offers a highly precise approximation of the principles governing biological neuron action potentials, earning them the Nobel Prize in Physiology or Medicine in 1963. Although the H-H model has high biological fidelity, it is computationally intensive. Other models, such as the Adaptive Exponential Integrate-and-Fire (aEIF) model [5] and the Izhikevich model [35], strive to balance biological fidelity and computational simplicity.

2.2 Computational Unit Of SNNs

In contemporary SNNs, neuron models predominantly rely on the LIF model. While the mathematical formulation of the LIF model involves a time-dependent differential equation, actual computer computations discretize this process for approximation.

$$H[t] = V[t-1] + \frac{1}{\tau} (X[t] - (V[t-1] - V_{reset})) \quad (2)$$

$$S[t] = \Theta(H[t] - V_{threshold}) \quad (3)$$

$$V[t] = H[t](1 - S[t]) + V_{reset}S[t] \quad (4)$$

$$\Theta(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (5)$$

where τ represents the membrane time constant, V_{reset} is the resting potential, $V_{threshold}$ is the threshold voltage, and $X[t]$, $H[t]$, $V[t]$ represent the input current, the membrane potential before and after spiking firing at time step t , respectively. The specific implementation of LIF neurons can vary [18].

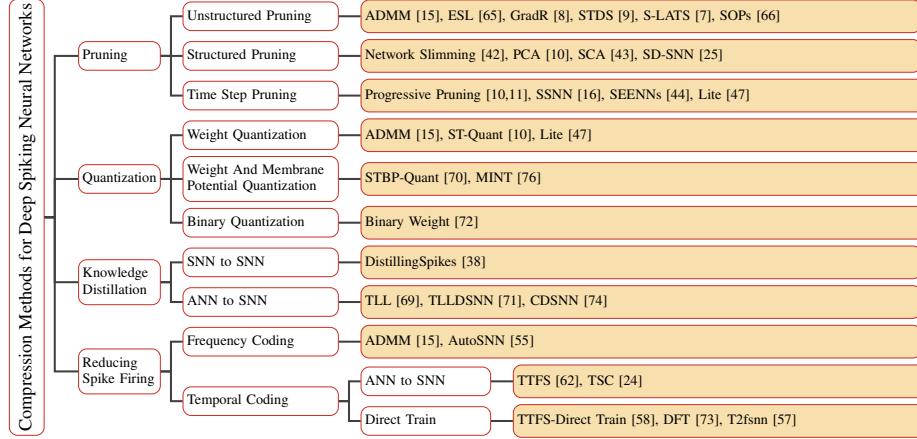


Fig. 1: Taxonomy of Model Compression methods for Deep Spiking Neural Networks.

3 Methods

DSNNs share many commonalities with DANNs, and some research has focused on these commonalities and proposed a unified pruning architecture [7]. Concepts like pruning, quantization, and knowledge distillation, initially developed for DANNs, can be adapted for DSNNs. However, the unique spiking and temporal characteristics of DSNNs often necessitate modifications to these techniques to ensure optimal performance. Additionally, DSNNs can be compressed in unique ways, such as leveraging spike sparsity, temporal coding, and pruning of time steps.

3.1 Pruning

DSNNs need to be deployed on neuromorphic hardware [77], which differs significantly from general-purpose computers in terms of operational logic. As a result, the value of pruning must be reassessed. Both unstructured and structured pruning are important for DSNNs, contrary to the inefficiency of unstructured pruning in DANNs [51], despite the scarcity of related work [46].

Unstructured Pruning. Unstructured pruning involves pruning weights or neurons.

ADMM-method [15] utilizes the Alternating Direction Method of Multipliers (ADMM) for connection pruning based on soft constraints, demonstrating effectiveness in reducing parameter memory space and baseline computational cost in DSNNs. It compresses connections, weight bit-widths, and spike frequencies simultaneously.

ESL [65] begins with a sparse network generated using the Erdős–Rényi random graph model, dynamically pruning weak connections and generating new ones according to structural plasticity rules. This approach, which updates the connection mask every *Titter* iterations and employs various growth methods, outperforms the ADMM-method.

GradR [8], inspired by synapse formation and elimination in the nervous system, redefines synaptic parameters using $w = sReLU(\theta)$, where s is determined at initialization and remains unchanged. It employs different training strategies for active and inactive connections, enhancing network exploration and recovery of dead neurons.

STDS [9] improves on GradR by using nonlinear reparameterization with $w = sign(\theta) \cdot (|\theta| - d)_+, d \geq 0$. When the size of the connection w is below threshold d , it is considered a filopodium, and the equivalent weight is zero. In any state, the parameters have gradients, allowing transitions between positive and negative weights, exploring the network space more fully. Pruning speed is adjusted by regulating the change in d .

S-LATS [7], a theoretical framework that reformulates soft threshold pruning as an implicit optimization problem and solves it using the Iterative Shrinkage-Thresholding Algorithm (ISTA), which is a classic method in the fields of sparse recovery and compressed sensing. It is proven that in the underlying optimization problem, the L1 coefficient is jointly determined by the threshold and the learning rate, allowing any threshold tuning strategy to be interpreted as a scheme for adjusting the L1 penalty. Through in-depth research on threshold scheduling based on the framework, an optimal threshold scheduler is derived, which maintains a stable L1 regularization coefficient, thereby providing a time-invariant objective function from an optimization perspective. A new family of pruning algorithms is proposed, including pruning during training, early pruning, and pruning at initialization, and get the best result both in DANNs and DSNNs.

SOPs-method [66] introduces an synaptic operation (SOP) metric (this paper define SOP as the operations performed when a spike passes through a synapse) to quantify power consumption in SNNs and uses an energy penalty term for energy-constrained unstructured weight and neuron pruning, maximizing efficiency through sparsity. During training, a binary mask m is reparameterized by α as $m = H(\alpha)$ and approximated with a scaled sigmoid function as $\lim_{\beta \rightarrow \infty} \sigma(\alpha; \beta) = \frac{1}{1+e^{-\beta\alpha}}$. Gradual scheduling methods adjust the scale factor β , to achieve different compressing rate.

Structured Pruning. Structured Pruning focuses on channel pruning for deep convolutional spiking neural networks.

Just the migration from DANNs Networks Slimming [48], DSNNs Networks Slimming [42] penalizes the scaling factors of the Batch-Normal layers, pruning channels with smaller scaling factors. However, due to the thresholding nature of DSNNs, the pruning is not thorough enough, achieving only 60% channel pruning while maintaining accuracy without much loss.

Principal Component Analysis(PCA) based pruning [10] targets redundant filters, a method also used in ANNs [21], adapted for SNNs. The pruning method is modified by using the principal component analysis of neurons' average cumulative membrane potentials to determine significant spatial dimensions for structured pruning. This step results in a 10-14 fold reduction in model size.

The network pruning framework based on Spike Channel Activity(SCA) [43] is inspired by synaptic plasticity mechanisms. During training, channels with lower average spike firing frequency are pruned, and convolutional kernels are dynamically adjusted

based on the gradients of the Batch-Normal layers scaling factors, regenerating those with larger gradients which is same to before work [17].

Adaptive structural development of SNN(SD-SNN) [25] is based on synaptic constraints inspired by dendritic spine plasticity. Synaptic constraints detect and remove significant redundancy in SNNs, and synapse regeneration effectively prevents and repairs excessive pruning. During the learning process, the neuron pruning rate and synapse regeneration rate are adaptively adjusted, leading to a stable SNN structure.

Time Step Pruning. Time step pruning is unique to SNNs due to their temporal dimension in data input.

In algorithms transfer DANNs to DSNNs, a high number of time steps is needed to accurately capture the values of the ReLU activation function, requiring 20-50 steps or more. This is unacceptable for SNNs in practical applications, as it introduces step times computational and memory burdens [3].

The main challenge of time step pruning is that when the number of time steps is too low, neurons in the latter parts of the model cannot fire, leading to vanishing gradients. This means back-propagation cannot occur, and the model cannot be trained.

A simple idea is progressive pruning [10,11]. First, train the model with a long time step to ensure a high spike rate in the final layer. Then, prune the time steps while the model can still produce spikes in the final layer even with fewer time steps, allowing back-propagation to proceed normally. Gradually reduce the time steps until reaching an extremely low number, potentially achieving good results even with a single time step. The test dataset used for this approach is a standard image dataset.

Shrinking SNN (SSNN) [16] addresses the issue of disappearing spikes by modifying the model structure, enabling the model to derive loss from the spikes of intermediate neurons, back-propagate, and train normally. This method has shown good results but is currently limited to the DVS dataset, demonstrating its compression capabilities on datasets with temporal sequences. It may also be extendable to static datasets.

Spiking Early-Exit Neural Networks (SEENNs) [44] focus on balancing efficiency and accuracy. This method treats the time step as a variable and filters uncertain predictions through a confidence threshold. For uncertain predictions, reinforcement learning is used to determine the appropriate number of time steps, achieving automatic time step adjustment. On the CIFAR-10 dataset, this method achieved an average of nearly one time step while maintaining higher accuracy compared to other methods with longer time step.

Lite [47] thoroughly explores the optimal number of time steps for each layer through neural architecture search, characterizes the energy consumption through the metric of total synaptic operations, and achieves a good balance between energy efficiency and accuracy by controlling the pruning strength with a penalty factor.

3.2 Quantization

Quantization transforms large numerical data into a discrete set of values, aiming to minimize bit usage while maintaining computational precision. In DSNNs, quantization

includes membrane potential quantization alongside weights, without activation quantization. This study categorizes quantization efforts into weight quantization and weight and membrane potential quantization, along with specialized binary quantization.

Weight Quantization. ADMM-Quant [15] employs the ADMM to enforce quantization constraints on weights, albeit with a limitation to weight-only uniform quantization.

ST-Quant [10] utilizes K-means clustering-based weight sharing quantization techniques to further compress the model, akin to the approach taken by [26] for weight quantization.

Lite [47] adopts Neural Architecture Search (NAS) to automatically discover suitable mixed-precision bit-widths, enabling different layers to adopt varying levels of quantization. This is integrated within a unified framework that concurrently searches for optimal bit-widths, time steps, and network architectures, guided by a penalty term incorporated into the loss function for joint training.

Weight And Membrane Potential Quantization. STBP-Quant [70] transforms high-precision floating-point computations in existing direct training algorithms to low-bitwidth integer operations, introducing Integer-STBP, an algorithm that facilitates training and inference of SNNs using solely integer arithmetic. This enables implementation on low-power edge devices for online learning and inference.

MINT [76], building upon STBP-Quant, eliminates the requirement for multipliers in conventional uniform quantization during inference by sharing scaling factors between weights and membrane potentials. Scaling factors are retained during training to ensure competitive accuracy, while their necessity is obviated during inference through shared scaling, thus removing 32-bit multipliers in hardware. By quantizing memory-intensive membrane potentials to extremely low precision (2-bits), significant reductions in memory usage are achieved.

Binary Quantization. A novel weight-threshold balancing transformation [72] is proposed, adjusting the threshold of spiking neurons to convert high-precision weights into binary values (-1 or 1), effectively yielding binary SNNs. This drastically reduces weight memory requirements, albeit accompanied by increased neuron threshold storage (originally all are same).

3.3 Knowledge Distillation

Knowledge distillation (KD) is a commonly used model compression method, which uses a large teacher model to guide the training of a small student model. Specifically, knowledge distillation takes the knowledge of the teacher model as a supervision signal during the training process, so that the student model not only learns from the data, but also receives guidance from the teacher model, thereby achieving better performance.

Knowledge distillation from SNN to SNN. As the scale of SNNs continues to expand, its demand for storage and computing resources also gradually increases, hindering its application in real life. To address this issue, Kushawaha et al. [38] proposed the first knowledge distillation method specially designed for SNNs to minimize loss of accuracy. They use the spiking activation tensors of the teacher and student models to simultaneously calculate the full loss and the sliding window loss as new loss functions. Then they froze the weights of the teacher SNN and trained the student SNN. Moreover, they also introduced a multi-stage distillation procedure to further improve the performance of student SNN. Experiments on three standard image classification datasets show that their method improves the performance of student SNN.

Knowledge distillation from ANN to SNN. Takuya et al. [69] proposed to use knowledge distillation of KL divergence to train low-latency SNN. They first utilized a large ANN as the teacher model to train a small ANN, and then converted the small ANN into a SNN. Finally, they distill knowledge from the large teacher ANN into the SNN and use approximate gradients to solve the problem of non-differentiable spikes. The experimental results on the CIFAR-100 dataset show that they achieved the lowest inference latency while maintaining accuracy.

Tran et al. [71] proposed a training technique to convert ANNs to SNNs which is able to learn more hidden information by using knowledge distillation. They combined knowledge distillation and batch normalization through time (BNTT) to improve the performance of converted SNNs and reduce its power consumption. Experiments on Tiny-ImageNet, CIFAR-10 and CIFAR-100 show that their method successfully improves accuracy and reduces inference latency.

Xu et al. [74] proposed a knowledge distillation training method combining ANN and SNN. They combined spike coding and joint loss function to solve the problem of non-differentiable SNN spikes. They proposed two knowledge distillation methods: response-based KD and feature-based KD, which extract knowledge from the last layer output of the teacher model and some intermediate layers of the teacher model respectively. Comprehensive experimental results demonstrate the effectiveness of their approach.

3.4 Reducing Firing Rate

In neuromorphic hardware, a common metric for characterizing the energy consumption is synaptic operations [13], and reducing the number of spikes can effectively decrease synaptic operations. Frequency coding is a commonly used encoding method, favored for its simplicity and efficiency, which is conducive to training; however, it typically requires a higher number of spikes to achieve satisfactory results under this encoding. Temporal coding, on the other hand, conveys the same information with fewer spikes, allowing at most a single spike per neuron in the neural network, effectively reducing the number of spikes during inference. The trade-off is longer time steps, which are typically used in the transition from ANN-SNN algorithms. Due to space constraints, we will only discuss the Temporal coding of ANN-SNN. Direct training temporal coding can be seen here [73,58,57].

Frequency Coding. ADMM-method [15] reduces spike firing frequency during training by imposing a penalty factor on the membrane potential. AutoSNN [55] is a direct training algorithm that adopts a one-shot weight-sharing approach based on an evolutionary algorithm to automatically search for and discover energy-efficient SNN architectures suitable for specific tasks. The article mentions that the global average pooling layer can reduce the energy efficiency of SNNs, so the maximum pooling layer is recommended.

Temporal Coding. Time-to-First-Spike(TTFS) coding [62] approximates the real-valued ReLU activation in DANNs to the delay of the first spike in the corresponding spike sequence in DSNNs, requiring at most one spike per activation. However, the cost is a longer time step, and the memory access and computational costs associated with long time steps remain high, as peak neurons need to track synapses and receive the first peak from synapses after each time step.

Temporal-SwitchCoding (TSC), and the corresponding TSC spiking neuron model [24], is presented better than TTFS. Each pixel of the input image is presented through two spikes, with the time interval between the two spikes being proportional to the pixel intensity. Throughout the inference process, each synapse performs at most two memory accesses and two addition operations, significantly improving the energy efficiency of SNNs.

4 Future Directions

Efficient Neuron Model. There are models that increase accuracy with more parameters and complex structures [27,75], as well as models that precisely model biological neurons. However, there is limited research on developing more efficient neuron models specifically for deep learning [54]. Two potential paths exist: one involves modeling neurons more finely but with fewer neurons in shallow networks, and the other involves modeling neurons more coarsely with a larger number of neurons in deep networks. Precise modeling of biological neurons requires substantial computational resources but offers greater representational capacity. By accurately simulating biological neurons, we can achieve complex network functions with fewer neurons [39,1,80], which may present an energy efficiency advantage.

Balancing biological fidelity with computational efficiency presents a trade-off. Exploring how to abstract better neuron characteristics for modeling is a key problem that warrants further investigation to realize efficient neural networks.

Unified Compression Architecture. DANNs and DSNNs share commonalities, and their similarities may represent the fundamental characteristics of neural networks. Methods applicable only to specific types of neural networks might not address the essence of neural networks. For instance, a unified framework for soft threshold pruning has been proposed, revealing the characteristics of neural network soft threshold pruning [7]. Similarly, the PCA method's analysis of convolutional kernel similarity followed by pruning demonstrates the effectiveness of pruning similar components

in neural networks [21,10]. The difficulties encountered in regularization pruning of Batch-Normal layer parameters suggest that a unified solution is needed [42,48].

Key issues in DSNNs compression include: What are the differences between DANNs and DSNNs during compression? Which DANNs methods can be transferred to DSNNs? If there are differences, how should they be modified? Is there a unified method?

Efficient Specialization Program. Are there more efficient encoding schemes? Time encoding requires long time steps, and frequency encoding requires higher firing frequencies. Fully utilizing the time and frequency information of spikes is an important issue. Existing work combines time and frequency, but can we go further? Moreover, we may need more specialized model structures and further research on them, beyond just minor modifications to VGG and ResNet like VGGSNN and Sew-ResNet [50,39,28].

Co-Design Compression With Hardware. The requirement for DSNNs to be deployed on neuromorphic hardware to gain advantages over DANNs affects the direction of DSNNs optimization. A variety of neuromorphic hardware platforms [13,59,2,56,63] may necessitate hardware-related collaborative compression work.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Beniaguev, D., Segev, I., London, M.: Single cortical neurons as deep artificial neural networks. *Neuron* **109**(17), 2727–2739 (2021)
2. Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.M., Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A., Boahen, K.: Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE* **102**(5), 699–716 (2014)
3. Bhattacharjee, A., Yin, R., Moitra, A., Panda, P.: Are snns truly energy-efficient? - a hardware perspective. *ArXiv* **abs/2309.03388** (2023)
4. Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., Beigne, E.: Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* **15**(2), 1–35 (2019)
5. Brette, R., Gerstner, W.: Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology* **94**(5), 3637–3642 (2005)
6. Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., Miao, Y.: Review of image classification algorithms based on convolutional neural networks. *Remote Sensing* **13**(22), 4712 (2021)
7. Chen, Y., Ma, Z., Fang, W., Zheng, X., Yu, Z., Tian, Y.: A unified framework for soft threshold pruning. *arXiv preprint arXiv:2302.13019* (2023)
8. Chen, Y., Yu, Z., Fang, W., Huang, T., Tian, Y.: Pruning of deep spiking neural networks through gradient rewiring. In: Zhou, Z.H. (ed.) *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. pp. 1713–1721. International Joint Conferences on Artificial Intelligence Organization (8 2021). <https://doi.org/10.24963/ijcai.2021/236>, main Track

9. Chen, Y., Yu, Z., Fang, W., Ma, Z., Huang, T., Tian, Y.: State transition of dendritic spines improves learning of sparse spiking neural networks. In: International Conference on Machine Learning. pp. 3701–3715. PMLR (2022)
10. Chowdhury, S.S., Garg, I., Roy, K.: Spatio-temporal pruning and quantization for low-latency spiking neural networks. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–9. IEEE (2021)
11. Chowdhury, S.S., Rathi, N., Roy, K.: Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning. In: European Conference on Computer Vision. pp. 709–726. Springer (2022)
12. Dampfhofer, M., Mesquida, T., Valentian, A., Anghel, L.: Backpropagation-based learning techniques for deep spiking neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems (2023)
13. Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al.: Loihi: A neuromorphic manycore processor with on-chip learning. Ieee Micro **38**(1), 82–99 (2018)
14. Dayan, P., Abbott, L.F.: A mathematical model of spiking neurons. In: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems (2001)
15. Deng, L., Wu, Y., Hu, Y., Liang, L., Li, G., Hu, X., Ding, Y., Li, P., Xie, Y.: Comprehensive snn compression using admm optimization and activity regularization. IEEE transactions on neural networks and learning systems **34**(6), 2791–2805 (2021)
16. Ding, Y., Zuo, L., Jing, M., He, P., Xiao, Y.: Shrinking your timestep: Towards low-latency neuromorphic object recognition with spiking neural network. Proc. AAAI Conf. Artif. Intell. (2024)
17. Evcı, U., Gale, T., Menick, J., Castro, P.S., Elsen, E.: Rigging the lottery: Making all tickets winners. In: International conference on machine learning. pp. 2943–2952. PMLR (2020)
18. Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., Huang, L., Zhou, H., Li, G., Tian, Y.: Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances **9**(40), eadi1480 (2023)
19. Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., Tian, Y.: Deep residual learning in spiking neural networks. In: Neural Information Processing Systems (2021)
20. Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**, 154–180 (2019)
21. Garg, I., Panda, P., Roy, K.: A low effort approach to structured cnn design using pca. IEEE Access **8**, 1347–1360 (2019)
22. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision, pp. 291–326. Chapman and Hall/CRC (2022)
23. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**(6), 1789–1819 (2021)
24. Han, B., Roy, K.: Deep spiking neural network: Energy efficiency through time based coding. In: European Conference on Computer Vision. pp. 388–404. Springer (2020)
25. Han, B., Zhao, F., Zeng, Y., Pan, W.: Adaptive sparse structure development with pruning and regeneration for spiking neural networks. arXiv preprint arXiv:2211.12219 (2022)
26. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
27. Hao, Z., Shi, X., Huang, Z., Bu, T., Yu, Z., Huang, T.: A progressive training framework for spiking neural networks with learnable multi-hierarchical model. In: The Twelfth International Conference on Learning Representations (2023)

28. Hasani, R., Lechner, M., Amini, A., Rus, D., Grosu, R.: Liquid time-constant networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 7657–7666 (2021)
29. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog. pp. 770–778 (2016)
30. He, Y., Xiao, L.: Structured pruning for deep convolutional neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **46**(5), 2900–2919 (2024). <https://doi.org/10.1109/TPAMI.2023.3334614>
31. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* **117**(4), 500 (1952)
32. Hu, Y.Z., Tang, H., Pan, G.: Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems* **34**, 5200–5205 (2021)
33. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. IEEE Conf. Comput. Vis. Pattern Recog. pp. 4700–4708 (2017)
34. Huang, L.W., Ma, Z., Yu, L., Zhou, H., Tian, Y.: Deep spiking neural networks with high representation similarity model visual pathways of macaque and mouse. In: AAAI Conference on Artificial Intelligence (2023)
35. Izhikevich, E.M.: Simple model of spiking neurons. *IEEE Transactions on neural networks* **14**(6), 1569–1572 (2003)
36. Kim, Y., Panda, P.: Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience* **15** (2020)
37. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **25** (2012)
38. Kushawaha, R.K., Kumar, S., Banerjee, B., Velmurugan, R.: Distilling spikes: Knowledge distillation in spiking neural networks. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 4536–4543. IEEE (2021)
39. Lechner, M., Hasani, R., Amini, A., Henzinger, T.A., Rus, D., Grosu, R.: Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence* **2**(10), 642–652 (2020)
40. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *nature* **521**(7553), 436–444 (2015)
41. Lee, C., Sarwar, S.S., Panda, P., Srinivasan, G., Roy, K.: Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience* **14** (2019)
42. Li, Y., Fang, X., Gao, Y., Zhou, D., Shen, J., Liu, J.K., Pan, G., Xu, Q.: Efficient structure slimming for spiking neural networks. *IEEE Transactions on Artificial Intelligence* (2024)
43. Li, Y., Shen, J., Xu, H., Chen, L., Pan, G., Zhang, Q., Xu, Q.: Towards efficient deep spiking neural networks construction with spiking activity based pruning. *Proc. Int. Conf. Mach. Learn.* (2024)
44. Li, Y., Geller, T., Kim, Y., Panda, P.: Seenn: Towards temporal spiking early exit neural networks. *Advances in Neural Information Processing Systems* **36** (2024)
45. Li, Y., Yin, R., Kim, Y., Panda, P.: Efficient human activity recognition with spatio-temporal spiking neural networks. *Frontiers in Neuroscience* **17** (2023)
46. Liu, B., Wang, M., Foroosh, H., Tappen, M., Pensky, M.: Sparse convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 806–814 (2015)
47. Liu, Q., Yan, J., Zhang, M., Pan, G., Li, H.: Lite-snn: Designing lightweight and efficient spiking neural network through spatial-temporal compressive network search and joint optimization. *arXiv preprint arXiv:2401.14652* (2024)
48. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision. pp. 2736–2744 (2017)
49. Lv, C., Xu, J., Zheng, X.: Spiking convolutional neural networks for text classification. In: International Conference on Learning Representations (2023)

50. Ma, G., Jiang, R., Yan, R., Tang, H.: Temporal conditioning spiking latent variable models of the neural response to natural visual scenes. *Advances in Neural Information Processing Systems* **36** (2024)
51. Ma, X., Lin, S., Ye, S., He, Z., Zhang, L., Yuan, G., Tan, S.H., Li, Z., Fan, D., Qian, X., Lin, X., Ma, K., Wang, Y.: Non-structured dnn weight pruning—is it beneficial in any platform? *IEEE Transactions on Neural Networks and Learning Systems* **33**(9), 4930–4944 (2022). <https://doi.org/10.1109/TNNLS.2021.3063265>
52. Maass, W.: Networks of spiking neurons: The third generation of neural network models. *Electron. Colloquium Comput. Complex.* **TR96** (1996)
53. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence* **44**(7), 3523–3542 (2021)
54. Moser, B.A., Lunglmayr, M.: Quantization in spiking neural networks (2024)
55. Na, B., Mok, J., Park, S., Lee, D., Choe, H., Yoon, S.: Autosnn: Towards energy-efficient spiking neural networks. In: *International Conference on Machine Learning*. pp. 16253–16269. PMLR (2022)
56. Painkras, E., Plana, L.A., Garside, J., Temple, S., Galluppi, F., Patterson, C., Lester, D.R., Brown, A.D., Furber, S.B.: Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits* **48**(8), 1943–1953 (2013)
57. Park, S., Kim, S., Na, B., Yoon, S.: T2fsnn: Deep spiking neural networks with time-to-first-spike coding. 2020 57th ACM/IEEE Design Automation Conference (DAC) pp. 1–6 (2020)
58. Park, S., Yoon, S.: Training energy-efficient deep spiking neural networks with time-to-first-spike coding. arXiv preprint arXiv:2106.02568 (2021)
59. Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al.: Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature* **572**(7767), 106–111 (2019)
60. Pfeiffer, M., Pfeil, T.: Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in neuroscience* **12**, 409662 (2018)
61. Roy, K., Jaiswal, A.R., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. *Nature* **575**, 607 – 617 (2019)
62. Rueckauer, B., Liu, S.C.: Conversion of analog to spiking neural networks using sparse temporal coding. In: *2018 IEEE international symposium on circuits and systems (ISCAS)*. pp. 1–5. IEEE (2018)
63. Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., Millner, S.: A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: *2010 ieee international symposium on circuits and systems (iscas)*. pp. 1947–1950. IEEE (2010)
64. Sengupta, A., Ye, Y., Wang, R.Y., Liu, C., Roy, K.: Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience* **13** (2018)
65. Shen, J., Xu, Q., Liu, J.K., Wang, Y., Pan, G., Tang, H.: Esl-snns: An evolutionary structure learning strategy for spiking neural networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 37, pp. 86–93 (2023)
66. Shi, X., Ding, J., Hao, Z., Yu, Z.: Towards energy efficient spiking neural networks: An unstructured pruning framework. In: *The Twelfth International Conference on Learning Representations* (2023)
67. Shi, X., Hao, Z., Yu, Z.: Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. ArXiv **abs/2403.14302** (2024)
68. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: *Proc. Int. Conf. Learn. Represent.* (2015)
69. Takuya, S., Zhang, R., Nakashima, Y.: Training low-latency spiking neural network through knowledge distillation. In: *2021 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*. pp. 1–3. IEEE (2021)

70. Tan, P.Y., Wu, C.W.: A low-bitwidth integer-stbp algorithm for efficient training and inference of spiking neural networks. In: Proceedings of the 28th Asia and South Pacific Design Automation Conference. pp. 651–656 (2023)
71. Tran, T.D., Le, K.T., Nguyen, A.L.T.: Training low-latency deep spiking neural networks with knowledge distillation and batch normalization through time. In: 2022 5th International Conference on Computational Intelligence and Networks (CINE). pp. 01–06. IEEE (2022)
72. Wang, Y., Xu, Y., Yan, R., Tang, H.: Deep spiking neural networks with binary weights for object recognition. *IEEE Transactions on Cognitive and Developmental Systems* **13**(3), 514–523 (2020)
73. Wei, W., Zhang, M., Qu, H., Belatreche, A., Zhang, J., Chen, H.: Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10552–10562 (2023)
74. Xu, Q., Li, Y., Shen, J., Liu, J.K., Tang, H., Pan, G.: Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7886–7895 (2023)
75. Yao, X., Li, F., Mo, Z., Cheng, J.: Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems* **35**, 32160–32171 (2022)
76. Yin, R., Li, Y., Moitra, A., Panda, P.: Mint: Multiplier-less integer quantization for energy efficient spiking neural networks. In: 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 830–835. IEEE (2024)
77. Yin, R., Moitra, A., Bhattacharjee, A., Kim, Y., Panda, P.: Sata: Sparsity-aware training accelerator for spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022)
78. Yin, W., Kann, K., Yu, M., Schütze, H.: Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923 (2017)
79. Zhang, H., Zhou, C., Yu, L., Huang, L., Ma, Z., Fan, X., Zhou, H., Tian, Y.: Sglformer: Spiking global-local-fusion transformer with high performance. *Frontiers in Neuroscience* **18** (2024)
80. Zhao, M., Wang, N., Jiang, X., Ma, X., Ma, H., He, G., Du, K., Ma, L., Huang, T.: Metaworm: An integrative data-driven model simulating *c. elegans* brain, body and environment interactions. bioRxiv (2024)
81. Zhou, C., Zhang, H., Yu, L., Ye, Y., Zhou, Z., Huang, L., Ma, Z., Fan, X., Zhou, H., Tian, Y.: Direct training high-performance deep spiking neural networks: A review of theories and methods. arXiv preprint arXiv:2405.04289 (2024)
82. Zhou, Z., Zhu, Y., He, C., Wang, Y., Yan, S., Tian, Y., Yuan, L.: Spikformer: When spiking neural network meets transformer. ArXiv **abs/2209.15425** (2022)
83. Zhu, Y., Yu, Z., Fang, W., Xie, X., Huang, T., Masquelier, T.: Training spiking neural networks with event-driven backpropagation. In: Neural Information Processing Systems (2022)
84. Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. *Proceedings of the IEEE* **111**(3), 257–276 (2023)